Package ‘paws.developer.tools’

August 24, 2021

Title 'Amazon Web Services' Developer Tools Services

Version 0.1.12

Description Interface to 'Amazon Web Services' developer tools services, including version control, continuous integration and deployment, and more <https://aws.amazon.com/products/developer-tools/>.

License Apache License (>= 2.0)

URL https://github.com/paws-r/paws

BugReports https://github.com/paws-r/paws/issues

Imports paws.common (>= 0.3.0)

Suggests testthat

Encoding UTF-8

RoxygenNote 7.1.1

Collate 'cloud9_service.R' 'cloud9_interfaces.R' 'cloud9_operations.R' 'codebuild_service.R' 'codebuild_interfaces.R' 'codebuild_operations.R' 'codecommit_service.R' 'codecommit_interfaces.R' 'codecommit_operations.R' 'codedeploy_service.R' 'codedeploy_interfaces.R' 'codedeploy_operations.R' 'codepipeline_service.R' 'codepipeline_interfaces.R' 'codepipeline_operations.R' 'codestar_service.R' 'codestar_interfaces.R' 'codestar_operations.R' 'xray_service.R' 'xray_interfaces.R' 'xray_operations.R'

NeedsCompilation no

Author David Kretch [aut, cre],
Adam Banker [aut],
Amazon.com, Inc. [cph]

Maintainer David Kretch <david.kretch@gmail.com>

Repository CRAN

Date/Publication 2021-08-24 19:30:06 UTC

R topics documented:

`cloud9` *AWS Cloud9*

Description

AWS Cloud9 is a collection of tools that you can use to code, build, run, test, debug, and release software in the cloud.

For more information about AWS Cloud9, see the AWS Cloud9 User Guide.

AWS Cloud9 supports these operations:

- `create_environment_ec2`: Creates an AWS Cloud9 development environment, launches an Amazon EC2 instance, and then connects from the instance to the environment.
- `create_environment_membership`: Adds an environment member to an environment.
- `delete_environment`: Deletes an environment. If an Amazon EC2 instance is connected to the environment, also terminates the instance.
- `delete_environment_membership`: Deletes an environment member from an environment.
- `describe_environment_memberships`: Gets information about environment members for an environment.
- `describe_environments`: Gets information about environments.
- `describe_environment_status`: Gets status information for an environment.
- `list_environments`: Gets a list of environment identifiers.
- `list_tags_for_resource`: Gets the tags for an environment.
- `tag_resource`: Adds tags to an environment.
- `untag_resource`: Removes tags from an environment.
- `update_environment`: Changes the settings of an existing environment.
- `update_environment_membership`: Changes the settings of an existing environment member for an environment.

Usage

```
cloud9(config = list())
```

Arguments

`config`	Optional configuration of credentials, endpoint, and/or region.

Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

Service syntax

```
svc <- cloud9(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

Operations

create_environment_ec2	Creates an AWS Cloud9 development environment, launches an Amazon Elastic Comp
create_environment_membership	Adds an environment member to an AWS Cloud9 development environment
delete_environment	Deletes an AWS Cloud9 development environment
delete_environment_membership	Deletes an environment member from an AWS Cloud9 development environment
describe_environment_memberships	Gets information about environment members for an AWS Cloud9 development enviro
describe_environments	Gets information about AWS Cloud9 development environments
describe_environment_status	Gets status information for an AWS Cloud9 development environment
list_environments	Gets a list of AWS Cloud9 development environment identifiers
list_tags_for_resource	Gets a list of the tags associated with an AWS Cloud9 development environment
tag_resource	Adds tags to an AWS Cloud9 development environment
untag_resource	Removes tags from an AWS Cloud9 development environment
update_environment	Changes the settings of an existing AWS Cloud9 development environment
update_environment_membership	Changes the settings of an existing environment member for an AWS Cloud9 developm

Examples

```
## Not run:
svc <- cloud9()
```

```
#
svc$create_environment_ec2(
  name = "my-demo-environment",
  automaticStopTimeMinutes = 60L,
  description = "This is my demonstration environment.",
  instanceType = "t2.micro",
  ownerArn = "arn:aws:iam::123456789012:user/MyDemoUser",
  subnetId = "subnet-1fab8aEX"
)

## End(Not run)
```

codebuild *AWS CodeBuild*

Description

AWS CodeBuild is a fully managed build service in the cloud. AWS CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy. AWS CodeBuild eliminates the need to provision, manage, and scale your own build servers. It provides prepackaged build environments for the most popular programming languages and build tools, such as Apache Maven, Gradle, and more. You can also fully customize build environments in AWS CodeBuild to use your own build tools. AWS CodeBuild scales automatically to meet peak build requests. You pay only for the build time you consume. For more information about AWS CodeBuild, see the *AWS CodeBuild User Guide*.

AWS CodeBuild supports these operations:

- `batch_delete_builds`: Deletes one or more builds.
- `batch_get_builds`: Gets information about one or more builds.
- `batch_get_projects`: Gets information about one or more build projects. A *build project* defines how AWS CodeBuild runs a build. This includes information such as where to get the source code to build, the build environment to use, the build commands to run, and where to store the build output. A *build environment* is a representation of operating system, programming language runtime, and tools that AWS CodeBuild uses to run a build. You can add tags to build projects to help manage your resources and costs.
- `batch_get_report_groups`: Returns an array of report groups.
- `batch_get_reports`: Returns an array of reports.
- `create_project`: Creates a build project.
- `create_report_group`: Creates a report group. A report group contains a collection of reports.
- `create_webhook`: For an existing AWS CodeBuild build project that has its source code stored in a GitHub or Bitbucket repository, enables AWS CodeBuild to start rebuilding the source code every time a code change is pushed to the repository.
- `delete_project`: Deletes a build project.

- delete_report: Deletes a report.
- delete_report_group: Deletes a report group.
- delete_resource_policy: Deletes a resource policy that is identified by its resource ARN.
- delete_source_credentials: Deletes a set of GitHub, GitHub Enterprise, or Bitbucket source credentials.
- delete_webhook: For an existing AWS CodeBuild build project that has its source code stored in a GitHub or Bitbucket repository, stops AWS CodeBuild from rebuilding the source code every time a code change is pushed to the repository.
- describe_test_cases: Returns a list of details about test cases for a report.
- get_resource_policy: Gets a resource policy that is identified by its resource ARN.
- import_source_credentials: Imports the source repository credentials for an AWS CodeBuild project that has its source code stored in a GitHub, GitHub Enterprise, or Bitbucket repository.
- invalidate_project_cache: Resets the cache for a project.
- list_builds: Gets a list of build IDs, with each build ID representing a single build.
- list_builds_for_project: Gets a list of build IDs for the specified build project, with each build ID representing a single build.
- list_curated_environment_images: Gets information about Docker images that are managed by AWS CodeBuild.
- list_projects: Gets a list of build project names, with each build project name representing a single build project.
- list_report_groups: Gets a list ARNs for the report groups in the current AWS account.
- list_reports: Gets a list ARNs for the reports in the current AWS account.
- list_reports_for_report_group: Returns a list of ARNs for the reports that belong to a `ReportGroup`.
- list_shared_projects: Gets a list of ARNs associated with projects shared with the current AWS account or user.
- list_shared_report_groups: Gets a list of ARNs associated with report groups shared with the current AWS account or user
- list_source_credentials: Returns a list of `SourceCredentialsInfo` objects. Each `SourceCredentialsInfo` object includes the authentication type, token ARN, and type of source provider for one set of credentials.
- put_resource_policy: Stores a resource policy for the ARN of a `Project` or `ReportGroup` object.
- start_build: Starts running a build.
- stop_build: Attempts to stop running a build.
- update_project: Changes the settings of an existing build project.
- update_report_group: Changes a report group.
- update_webhook: Changes the settings of an existing webhook.

Usage

```
codebuild(config = list())
```

Arguments

config	Optional configuration of credentials, endpoint, and/or region.

Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

Service syntax

```
svc <- codebuild(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

Operations

batch_delete_builds	Deletes one or more builds
batch_get_build_batches	Retrieves information about one or more batch builds
batch_get_builds	Gets information about one or more builds
batch_get_projects	Gets information about one or more build projects
batch_get_report_groups	Returns an array of report groups
batch_get_reports	Returns an array of reports
create_project	Creates a build project
create_report_group	Creates a report group
create_webhook	For an existing AWS CodeBuild build project that has its source code stored in a GitHub
delete_build_batch	Deletes a batch build
delete_project	Deletes a build project
delete_report	Deletes a report
delete_report_group	Deletes a report group
delete_resource_policy	Deletes a resource policy that is identified by its resource ARN
delete_source_credentials	Deletes a set of GitHub, GitHub Enterprise, or Bitbucket source credentials
delete_webhook	For an existing AWS CodeBuild build project that has its source code stored in a GitHub

describe_code_coverages	Retrieves one or more code coverage reports
describe_test_cases	Returns a list of details about test cases for a report
get_report_group_trend	Get report group trend
get_resource_policy	Gets a resource policy that is identified by its resource ARN
import_source_credentials	Imports the source repository credentials for an AWS CodeBuild project that has its sourc
invalidate_project_cache	Resets the cache for a project
list_build_batches	Retrieves the identifiers of your build batches in the current region
list_build_batches_for_project	Retrieves the identifiers of the build batches for a specific project
list_builds	Gets a list of build IDs, with each build ID representing a single build
list_builds_for_project	Gets a list of build IDs for the specified build project, with each build ID representing a si
list_curated_environment_images	Gets information about Docker images that are managed by AWS CodeBuild
list_projects	Gets a list of build project names, with each build project name representing a single buil
list_report_groups	Gets a list ARNs for the report groups in the current AWS account
list_reports	Returns a list of ARNs for the reports in the current AWS account
list_reports_for_report_group	Returns a list of ARNs for the reports that belong to a ReportGroup
list_shared_projects	Gets a list of projects that are shared with other AWS accounts or users
list_shared_report_groups	Gets a list of report groups that are shared with other AWS accounts or users
list_source_credentials	Returns a list of SourceCredentialsInfo objects
put_resource_policy	Stores a resource policy for the ARN of a Project or ReportGroup object
retry_build	Restarts a build
retry_build_batch	Restarts a failed batch build
start_build	Starts running a build
start_build_batch	Starts a batch build for a project
stop_build	Attempts to stop running a build
stop_build_batch	Stops a running batch build
update_project	Changes the settings of a build project
update_report_group	Updates a report group
update_webhook	Updates the webhook associated with an AWS CodeBuild build project

Examples

```
## Not run:
svc <- codebuild()
# The following example gets information about builds with the specified
# build IDs.
svc$batch_get_builds(
  ids = list(
    "codebuild-demo-project:9b0ac37f-d19e-4254-9079-f47e9a389eEX",
    "codebuild-demo-project:b79a46f7-1473-4636-a23f-da9c45c208EX"
  )
)

## End(Not run)
```

codecommit *AWS CodeCommit*

Description

This is the *AWS CodeCommit API Reference*. This reference provides descriptions of the operations and data types for AWS CodeCommit API along with usage examples.

You can use the AWS CodeCommit API to work with the following objects:

Repositories, by calling the following:

- `batch_get_repositories`, which returns information about one or more repositories associated with your AWS account.
- `create_repository`, which creates an AWS CodeCommit repository.
- `delete_repository`, which deletes an AWS CodeCommit repository.
- `get_repository`, which returns information about a specified repository.
- `list_repositories`, which lists all AWS CodeCommit repositories associated with your AWS account.
- `update_repository_description`, which sets or updates the description of the repository.
- `update_repository_name`, which changes the name of the repository. If you change the name of a repository, no other users of that repository can access it until you send them the new HTTPS or SSH URL to use.

Branches, by calling the following:

- `create_branch`, which creates a branch in a specified repository.
- `delete_branch`, which deletes the specified branch in a repository unless it is the default branch.
- `get_branch`, which returns information about a specified branch.
- `list_branches`, which lists all branches for a specified repository.
- `update_default_branch`, which changes the default branch for a repository.

Files, by calling the following:

- `delete_file`, which deletes the content of a specified file from a specified branch.
- `get_blob`, which returns the base-64 encoded content of an individual Git blob object in a repository.
- `get_file`, which returns the base-64 encoded content of a specified file.
- `get_folder`, which returns the contents of a specified folder or directory.
- `put_file`, which adds or modifies a single file in a specified repository and branch.

Commits, by calling the following:

- `batch_get_commits`, which returns information about one or more commits in a repository.
- `create_commit`, which creates a commit for changes to a repository.

- `get_commit`, which returns information about a commit, including commit messages and author and committer information.
- `get_differences`, which returns information about the differences in a valid commit specifier (such as a branch, tag, HEAD, commit ID, or other fully qualified reference).

Merges, by calling the following:

- `batch_describe_merge_conflicts`, which returns information about conflicts in a merge between commits in a repository.
- `create_unreferenced_merge_commit`, which creates an unreferenced commit between two branches or commits for the purpose of comparing them and identifying any potential conflicts.
- `describe_merge_conflicts`, which returns information about merge conflicts between the base, source, and destination versions of a file in a potential merge.
- `get_merge_commit`, which returns information about the merge between a source and destination commit.
- `get_merge_conflicts`, which returns information about merge conflicts between the source and destination branch in a pull request.
- `get_merge_options`, which returns information about the available merge options between two branches or commit specifiers.
- `merge_branches_by_fast_forward`, which merges two branches using the fast-forward merge option.
- `merge_branches_by_squash`, which merges two branches using the squash merge option.
- `merge_branches_by_three_way`, which merges two branches using the three-way merge option.

Pull requests, by calling the following:

- `create_pull_request`, which creates a pull request in a specified repository.
- `create_pull_request_approval_rule`, which creates an approval rule for a specified pull request.
- `delete_pull_request_approval_rule`, which deletes an approval rule for a specified pull request.
- `describe_pull_request_events`, which returns information about one or more pull request events.
- `evaluate_pull_request_approval_rules`, which evaluates whether a pull request has met all the conditions specified in its associated approval rules.
- `get_comments_for_pull_request`, which returns information about comments on a specified pull request.
- `get_pull_request`, which returns information about a specified pull request.
- `get_pull_request_approval_states`, which returns information about the approval states for a specified pull request.
- `get_pull_request_override_state`, which returns information about whether approval rules have been set aside (overriden) for a pull request, and if so, the Amazon Resource Name (ARN) of the user or identity that overrode the rules and their requirements for the pull request.

- list_pull_requests, which lists all pull requests for a repository.
- merge_pull_request_by_fast_forward, which merges the source destination branch of a pull request into the specified destination branch for that pull request using the fast-forward merge option.
- merge_pull_request_by_squash, which merges the source destination branch of a pull request into the specified destination branch for that pull request using the squash merge option.
- merge_pull_request_by_three_way. which merges the source destination branch of a pull request into the specified destination branch for that pull request using the three-way merge option.
- override_pull_request_approval_rules, which sets aside all approval rule requirements for a pull request.
- post_comment_for_pull_request, which posts a comment to a pull request at the specified line, file, or request.
- update_pull_request_approval_rule_content, which updates the structure of an approval rule for a pull request.
- update_pull_request_approval_state, which updates the state of an approval on a pull request.
- update_pull_request_description, which updates the description of a pull request.
- update_pull_request_status, which updates the status of a pull request.
- update_pull_request_title, which updates the title of a pull request.

Approval rule templates, by calling the following:

- associate_approval_rule_template_with_repository, which associates a template with a specified repository. After the template is associated with a repository, AWS CodeCommit creates approval rules that match the template conditions on every pull request created in the specified repository.
- batch_associate_approval_rule_template_with_repositories, which associates a template with one or more specified repositories. After the template is associated with a repository, AWS CodeCommit creates approval rules that match the template conditions on every pull request created in the specified repositories.
- batch_disassociate_approval_rule_template_from_repositories, which removes the association between a template and specified repositories so that approval rules based on the template are not automatically created when pull requests are created in those repositories.
- create_approval_rule_template, which creates a template for approval rules that can then be associated with one or more repositories in your AWS account.
- delete_approval_rule_template, which deletes the specified template. It does not remove approval rules on pull requests already created with the template.
- disassociate_approval_rule_template_from_repository, which removes the association between a template and a repository so that approval rules based on the template are not automatically created when pull requests are created in the specified repository.
- get_approval_rule_template, which returns information about an approval rule template.
- list_approval_rule_templates, which lists all approval rule templates in the AWS Region in your AWS account.

- list_associated_approval_rule_templates_for_repository, which lists all approval rule templates that are associated with a specified repository.
- list_repositories_for_approval_rule_template, which lists all repositories associated with the specified approval rule template.
- update_approval_rule_template_description, which updates the description of an approval rule template.
- update_approval_rule_template_name, which updates the name of an approval rule template.
- update_approval_rule_template_content, which updates the content of an approval rule template.

Comments in a repository, by calling the following:

- delete_comment_content, which deletes the content of a comment on a commit in a repository.
- get_comment, which returns information about a comment on a commit.
- get_comment_reactions, which returns information about emoji reactions to comments.
- get_comments_for_compared_commit, which returns information about comments on the comparison between two commit specifiers in a repository.
- post_comment_for_compared_commit, which creates a comment on the comparison between two commit specifiers in a repository.
- post_comment_reply, which creates a reply to a comment.
- put_comment_reaction, which creates or updates an emoji reaction to a comment.
- update_comment, which updates the content of a comment on a commit in a repository.

Tags used to tag resources in AWS CodeCommit (not Git tags), by calling the following:

- list_tags_for_resource, which gets information about AWS tags for a specified Amazon Resource Name (ARN) in AWS CodeCommit.
- tag_resource, which adds or updates tags for a resource in AWS CodeCommit.
- untag_resource, which removes tags for a resource in AWS CodeCommit.

Triggers, by calling the following:

- get_repository_triggers, which returns information about triggers configured for a repository.
- put_repository_triggers, which replaces all triggers for a repository and can be used to create or delete triggers.
- test_repository_triggers, which tests the functionality of a repository trigger by sending data to the trigger target.

For information about how to use AWS CodeCommit, see the AWS CodeCommit User Guide.

Usage

```
codecommit(config = list())
```

Arguments

`config`	Optional configuration of credentials, endpoint, and/or region.

Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

Service syntax

```
svc <- codecommit(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

Operations

associate_approval_rule_template_with_repository	Creates an association between an approval rule template and
batch_associate_approval_rule_template_with_repositories	Creates an association between an approval rule template and
batch_describe_merge_conflicts	Returns information about one or more merge conflicts in the
batch_disassociate_approval_rule_template_from_repositories	Removes the association between an approval rule template a
batch_get_commits	Returns information about the contents of one or more comm
batch_get_repositories	Returns information about one or more repositories
create_approval_rule_template	Creates a template for approval rules that can then be associa
create_branch	Creates a branch in a repository and points the branch to a co
create_commit	Creates a commit for a repository on the tip of a specified bra
create_pull_request	Creates a pull request in the specified repository
create_pull_request_approval_rule	Creates an approval rule for a pull request
create_repository	Creates a new, empty repository
create_unreferenced_merge_commit	Creates an unreferenced commit that represents the result of
delete_approval_rule_template	Deletes a specified approval rule template
delete_branch	Deletes a branch from a repository, unless that branch is the
delete_comment_content	Deletes the content of a comment made on a change, file, or
delete_file	Deletes a specified file from a specified branch
delete_pull_request_approval_rule	Deletes an approval rule from a specified pull request
delete_repository	Deletes a repository
describe_merge_conflicts	Returns information about one or more merge conflicts in the

describe_pull_request_events	Returns information about one or more pull request events
disassociate_approval_rule_template_from_repository	Removes the association between a template and a repository
evaluate_pull_request_approval_rules	Evaluates whether a pull request has met all the conditions sp
get_approval_rule_template	Returns information about a specified approval rule template
get_blob	Returns the base-64 encoded content of an individual blob in
get_branch	Returns information about a repository branch, including its
get_comment	Returns the content of a comment made on a change, file, or
get_comment_reactions	Returns information about reactions to a specified comment I
get_comments_for_compared_commit	Returns information about comments made on the compariso
get_comments_for_pull_request	Returns comments made on a pull request
get_commit	Returns information about a commit, including commit mess
get_differences	Returns information about the differences in a valid commit
get_file	Returns the base-64 encoded contents of a specified file and i
get_folder	Returns the contents of a specified folder in a repository
get_merge_commit	Returns information about a specified merge commit
get_merge_conflicts	Returns information about merge conflicts between the befor
get_merge_options	Returns information about the merge options available for m
get_pull_request	Gets information about a pull request in a specified repositor
get_pull_request_approval_states	Gets information about the approval states for a specified pul
get_pull_request_override_state	Returns information about whether approval rules have been
get_repository	Returns information about a repository
get_repository_triggers	Gets information about triggers configured for a repository
list_approval_rule_templates	Lists all approval rule templates in the specified AWS Regio
list_associated_approval_rule_templates_for_repository	Lists all approval rule templates that are associated with a sp
list_branches	Gets information about one or more branches in a repository
list_pull_requests	Returns a list of pull requests for a specified repository
list_repositories	Gets information about one or more repositories
list_repositories_for_approval_rule_template	Lists all repositories associated with the specified approval ru
list_tags_for_resource	Gets information about AWS tags for a specified Amazon Re
merge_branches_by_fast_forward	Merges two branches using the fast-forward merge strategy
merge_branches_by_squash	Merges two branches using the squash merge strategy
merge_branches_by_three_way	Merges two specified branches using the three-way merge str
merge_pull_request_by_fast_forward	Attempts to merge the source commit of a pull request into th
merge_pull_request_by_squash	Attempts to merge the source commit of a pull request into th
merge_pull_request_by_three_way	Attempts to merge the source commit of a pull request into th
override_pull_request_approval_rules	Sets aside (overrides) all approval rule requirements for a spe
post_comment_for_compared_commit	Posts a comment on the comparison between two commits
post_comment_for_pull_request	Posts a comment on a pull request
post_comment_reply	Posts a comment in reply to an existing comment on a compa
put_comment_reaction	Adds or updates a reaction to a specified comment for the use
put_file	Adds or updates a file in a branch in an AWS CodeCommit r
put_repository_triggers	Replaces all triggers for a repository
tag_resource	Adds or updates tags for a resource in AWS CodeCommit
test_repository_triggers	Tests the functionality of repository triggers by sending infor
untag_resource	Removes tags for a resource in AWS CodeCommit
update_approval_rule_template_content	Updates the content of an approval rule template
update_approval_rule_template_description	Updates the description for a specified approval rule template
update_approval_rule_template_name	Updates the name of a specified approval rule template

update_comment	Replaces the contents of a comment
update_default_branch	Sets or changes the default branch name for the specified rep
update_pull_request_approval_rule_content	Updates the structure of an approval rule created specifically
update_pull_request_approval_state	Updates the state of a user's approval on a pull request
update_pull_request_description	Replaces the contents of the description of a pull request
update_pull_request_status	Updates the status of a pull request
update_pull_request_title	Replaces the title of a pull request
update_repository_description	Sets or changes the comment or description for a repository
update_repository_name	Renames a repository

Examples

```
## Not run:
svc <- codecommit()
svc$associate_approval_rule_template_with_repository(
  Foo = 123
)

## End(Not run)
```

---

codedeploy *AWS CodeDeploy*

Description

AWS CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances running in your own facility, serverless AWS Lambda functions, or applications in an Amazon ECS service.

You can deploy a nearly unlimited variety of application content, such as an updated Lambda function, updated applications in an Amazon ECS service, code, web and configuration files, executables, packages, scripts, multimedia files, and so on. AWS CodeDeploy can deploy application content stored in Amazon S3 buckets, GitHub repositories, or Bitbucket repositories. You do not need to make changes to your existing code before you can use AWS CodeDeploy.

AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications, without many of the risks associated with error-prone manual deployments.

AWS CodeDeploy Components

Use the information in this guide to help you work with the following AWS CodeDeploy components:

- **Application**: A name that uniquely identifies the application you want to deploy. AWS CodeDeploy uses this name, which functions as a container, to ensure the correct combination of revision, deployment configuration, and deployment group are referenced during a deployment.

- **Deployment group**: A set of individual instances, CodeDeploy Lambda deployment configuration settings, or an Amazon ECS service and network details. A Lambda deployment group specifies how to route traffic to a new version of a Lambda function. An Amazon ECS deployment group specifies the service created in Amazon ECS to deploy, a load balancer, and a listener to reroute production traffic to an updated containerized application. An EC2/On-premises deployment group contains individually tagged instances, Amazon EC2 instances in Amazon EC2 Auto Scaling groups, or both. All deployment groups can specify optional trigger, alarm, and rollback settings.
- **Deployment configuration**: A set of deployment rules and deployment success and failure conditions used by AWS CodeDeploy during a deployment.
- **Deployment**: The process and the components used when updating a Lambda function, a containerized application in an Amazon ECS service, or of installing content on one or more instances.
- **Application revisions**: For an AWS Lambda deployment, this is an AppSpec file that specifies the Lambda function to be updated and one or more functions to validate deployment lifecycle events. For an Amazon ECS deployment, this is an AppSpec file that specifies the Amazon ECS task definition, container, and port where production traffic is rerouted. For an EC2/On-premises deployment, this is an archive file that contains source content—source code, webpages, executable files, and deployment scripts—along with an AppSpec file. Revisions are stored in Amazon S3 buckets or GitHub repositories. For Amazon S3, a revision is uniquely identified by its Amazon S3 object key and its ETag, version, or both. For GitHub, a revision is uniquely identified by its commit ID.

This guide also contains information to help you get details about the instances in your deployments, to make on-premises instances available for AWS CodeDeploy deployments, to get details about a Lambda function deployment, and to get details about Amazon ECS service deployments.

AWS CodeDeploy Information Resources

- AWS CodeDeploy User Guide
- AWS CodeDeploy API Reference Guide
- AWS CLI Reference for AWS CodeDeploy
- AWS CodeDeploy Developer Forum

Usage

```
codedeploy(config = list())
```

Arguments

`config`	Optional configuration of credentials, endpoint, and/or region.

Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

Service syntax

```
svc <- codedeploy(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

Operations

add_tags_to_on_premises_instances	Adds tags to on-premises instances
batch_get_application_revisions	Gets information about one or more application revisions
batch_get_applications	Gets information about one or more applications
batch_get_deployment_groups	Gets information about one or more deployment groups
batch_get_deployment_instances	This method works, but is deprecated
batch_get_deployments	Gets information about one or more deployments
batch_get_deployment_targets	Returns an array of one or more targets associated with a deployment
batch_get_on_premises_instances	Gets information about one or more on-premises instances
continue_deployment	For a blue/green deployment, starts the process of rerouting traffic from instanc
create_application	Creates an application
create_deployment	Deploys an application revision through the specified deployment group
create_deployment_config	Creates a deployment configuration
create_deployment_group	Creates a deployment group to which application revisions are deployed
delete_application	Deletes an application
delete_deployment_config	Deletes a deployment configuration
delete_deployment_group	Deletes a deployment group
delete_git_hub_account_token	Deletes a GitHub account connection
delete_resources_by_external_id	Deletes resources linked to an external ID
deregister_on_premises_instance	Deregisters an on-premises instance
get_application	Gets information about an application
get_application_revision	Gets information about an application revision
get_deployment	Gets information about a deployment
get_deployment_config	Gets information about a deployment configuration
get_deployment_group	Gets information about a deployment group
get_deployment_instance	Gets information about an instance as part of a deployment
get_deployment_target	Returns information about a deployment target
get_on_premises_instance	Gets information about an on-premises instance
list_application_revisions	Lists information about revisions for an application
list_applications	Lists the applications registered with the IAM user or AWS account
list_deployment_configs	Lists the deployment configurations with the IAM user or AWS account

list_deployment_groups	Lists the deployment groups for an application registered with the IAM user or
list_deployment_instances	The newer BatchGetDeploymentTargets should be used instead because it work
list_deployments	Lists the deployments in a deployment group for an application registered with
list_deployment_targets	Returns an array of target IDs that are associated a deployment
list_git_hub_account_token_names	Lists the names of stored connections to GitHub accounts
list_on_premises_instances	Gets a list of names for one or more on-premises instances
list_tags_for_resource	Returns a list of tags for the resource identified by a specified Amazon Resourc
put_lifecycle_event_hook_execution_status	Sets the result of a Lambda validation function
register_application_revision	Registers with AWS CodeDeploy a revision for the specified application
register_on_premises_instance	Registers an on-premises instance
remove_tags_from_on_premises_instances	Removes one or more tags from one or more on-premises instances
skip_wait_time_for_instance_termination	In a blue/green deployment, overrides any specified wait time and starts termina
stop_deployment	Attempts to stop an ongoing deployment
tag_resource	Associates the list of tags in the input Tags parameter with the resource identifie
untag_resource	Disassociates a resource from a list of tags
update_application	Changes the name of an application
update_deployment_group	Changes information about a deployment group

Examples

```
## Not run:
svc <- codedeploy()
svc$add_tags_to_on_premises_instances(
  Foo = 123
)

## End(Not run)
```

codepipeline *AWS CodePipeline*

Description

Overview

This is the AWS CodePipeline API Reference. This guide provides descriptions of the actions and data types for AWS CodePipeline. Some functionality for your pipeline can only be configured through the API. For more information, see the AWS CodePipeline User Guide.

You can use the AWS CodePipeline API to work with pipelines, stages, actions, and transitions.

Pipelines are models of automated release processes. Each pipeline is uniquely named, and consists of stages, actions, and transitions.

You can work with pipelines by calling:

- `create_pipeline`, which creates a uniquely named pipeline.

- `delete_pipeline`, which deletes the specified pipeline.
- `get_pipeline`, which returns information about the pipeline structure and pipeline metadata, including the pipeline Amazon Resource Name (ARN).
- `get_pipeline_execution`, which returns information about a specific execution of a pipeline.
- `get_pipeline_state`, which returns information about the current state of the stages and actions of a pipeline.
- `list_action_executions`, which returns action-level details for past executions. The details include full stage and action-level details, including individual action duration, status, any errors that occurred during the execution, and input and output artifact location details.
- `list_pipelines`, which gets a summary of all of the pipelines associated with your account.
- `list_pipeline_executions`, which gets a summary of the most recent executions for a pipeline.
- `start_pipeline_execution`, which runs the most recent revision of an artifact through the pipeline.
- `stop_pipeline_execution`, which stops the specified pipeline execution from continuing through the pipeline.
- `update_pipeline`, which updates a pipeline with edits or changes to the structure of the pipeline.

Pipelines include *stages*. Each stage contains one or more actions that must complete before the next stage begins. A stage results in success or failure. If a stage fails, the pipeline stops at that stage and remains stopped until either a new version of an artifact appears in the source location, or a user takes action to rerun the most recent artifact through the pipeline. You can call `get_pipeline_state`, which displays the status of a pipeline, including the status of stages in the pipeline, or `get_pipeline`, which returns the entire structure of the pipeline, including the stages of that pipeline. For more information about the structure of stages and actions, see AWS CodePipeline Pipeline Structure Reference.

Pipeline stages include *actions* that are categorized into categories such as source or build actions performed in a stage of a pipeline. For example, you can use a source action to import artifacts into a pipeline from a source such as Amazon S3. Like stages, you do not work with actions directly in most cases, but you do define and interact with actions when working with pipeline operations such as `create_pipeline` and `get_pipeline_state`. Valid action categories are:

- Source
- Build
- Test
- Deploy
- Approval
- Invoke

Pipelines also include *transitions*, which allow the transition of artifacts from one stage to the next in a pipeline after the actions in one stage complete.

You can work with transitions by calling:

- `disable_stage_transition`, which prevents artifacts from transitioning to the next stage in a pipeline.

- `enable_stage_transition`, which enables transition of artifacts between stages in a pipeline.

Using the API to integrate with AWS CodePipeline

For third-party integrators or developers who want to create their own integrations with AWS CodePipeline, the expected sequence varies from the standard API user. To integrate with AWS CodePipeline, developers need to work with the following items:

Jobs, which are instances of an action. For example, a job for a source action might import a revision of an artifact from a source.

You can work with jobs by calling:

- `acknowledge_job`, which confirms whether a job worker has received the specified job.
- `get_job_details`, which returns the details of a job.
- `poll_for_jobs`, which determines whether there are any jobs to act on.
- `put_job_failure_result`, which provides details of a job failure.
- `put_job_success_result`, which provides details of a job success.

Third party jobs, which are instances of an action created by a partner action and integrated into AWS CodePipeline. Partner actions are created by members of the AWS Partner Network.

You can work with third party jobs by calling:

- `acknowledge_third_party_job`, which confirms whether a job worker has received the specified job.
- `get_third_party_job_details`, which requests the details of a job for a partner action.
- `poll_for_third_party_jobs`, which determines whether there are any jobs to act on.
- `put_third_party_job_failure_result`, which provides details of a job failure.
- `put_third_party_job_success_result`, which provides details of a job success.

Usage

```
codepipeline(config = list())
```

Arguments

`config`	Optional configuration of credentials, endpoint, and/or region.

Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

Service syntax

```
svc <- codepipeline(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

Operations

acknowledge_job	Returns information about a specified job and whether that job has been received by t
acknowledge_third_party_job	Confirms a job worker has received the specified job
create_custom_action_type	Creates a new custom action that can be used in all pipelines associated with the AW
create_pipeline	Creates a pipeline
delete_custom_action_type	Marks a custom action as deleted
delete_pipeline	Deletes the specified pipeline
delete_webhook	Deletes a previously created webhook by name
deregister_webhook_with_third_party	Removes the connection between the webhook that was created by CodePipeline and
disable_stage_transition	Prevents artifacts in a pipeline from transitioning to the next stage in the pipeline
enable_stage_transition	Enables artifacts in a pipeline to transition to a stage in a pipeline
get_job_details	Returns information about a job
get_pipeline	Returns the metadata, structure, stages, and actions of a pipeline
get_pipeline_execution	Returns information about an execution of a pipeline, including details about artifacts
get_pipeline_state	Returns information about the state of a pipeline, including the stages and actions
get_third_party_job_details	Requests the details of a job for a third party action
list_action_executions	Lists the action executions that have occurred in a pipeline
list_action_types	Gets a summary of all AWS CodePipeline action types associated with your account
list_pipeline_executions	Gets a summary of the most recent executions for a pipeline
list_pipelines	Gets a summary of all of the pipelines associated with your account
list_tags_for_resource	Gets the set of key-value pairs (metadata) that are used to manage the resource
list_webhooks	Gets a listing of all the webhooks in this AWS Region for this account
poll_for_jobs	Returns information about any jobs for AWS CodePipeline to act on
poll_for_third_party_jobs	Determines whether there are any third party jobs for a job worker to act on
put_action_revision	Provides information to AWS CodePipeline about new revisions to a source
put_approval_result	Provides the response to a manual approval request to AWS CodePipeline
put_job_failure_result	Represents the failure of a job as returned to the pipeline by a job worker
put_job_success_result	Represents the success of a job as returned to the pipeline by a job worker
put_third_party_job_failure_result	Represents the failure of a third party job as returned to the pipeline by a job worker
put_third_party_job_success_result	Represents the success of a third party job as returned to the pipeline by a job worker
put_webhook	Defines a webhook and returns a unique webhook URL generated by CodePipeline

register_webhook_with_third_party	Configures a connection between the webhook that was created and the external tool
retry_stage_execution	Resumes the pipeline execution by retrying the last failed actions in a stage
start_pipeline_execution	Starts the specified pipeline
stop_pipeline_execution	Stops the specified pipeline execution
tag_resource	Adds to or modifies the tags of the given resource
untag_resource	Removes tags from an AWS resource
update_pipeline	Updates a specified pipeline with edits or changes to its structure

Examples

```
## Not run:
svc <- codepipeline()
svc$acknowledge_job(
  Foo = 123
)

## End(Not run)
```

`codestar` *AWS CodeStar*

Description

This is the API reference for AWS CodeStar. This reference provides descriptions of the operations and data types for the AWS CodeStar API along with usage examples.

You can use the AWS CodeStar API to work with:

Projects and their resources, by calling the following:

- `delete_project`, which deletes a project.
- `describe_project`, which lists the attributes of a project.
- `list_projects`, which lists all projects associated with your AWS account.
- `list_resources`, which lists the resources associated with a project.
- `list_tags_for_project`, which lists the tags associated with a project.
- `tag_project`, which adds tags to a project.
- `untag_project`, which removes tags from a project.
- `update_project`, which updates the attributes of a project.

Teams and team members, by calling the following:

- `associate_team_member`, which adds an IAM user to the team for a project.
- `disassociate_team_member`, which removes an IAM user from the team for a project.

- list_team_members, which lists all the IAM users in the team for a project, including their roles and attributes.
- update_team_member, which updates a team member's attributes in a project.

Users, by calling the following:

- create_user_profile, which creates a user profile that contains data associated with the user across all projects.
- delete_user_profile, which deletes all user profile information across all projects.
- describe_user_profile, which describes the profile of a user.
- list_user_profiles, which lists all user profiles.
- update_user_profile, which updates the profile for a user.

Usage

```
codestar(config = list())
```

Arguments

config	Optional configuration of credentials, endpoint, and/or region.

Value

A client for the service. You can call the service's operations using syntax like svc$operation(...), where svc is the name you've assigned to the client. The available operations are listed in the Operations section.

Service syntax

```
svc <- codestar(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

Operations

associate_team_member	Adds an IAM user to the team for an AWS CodeStar project
create_project	Creates a project, including project resources
create_user_profile	Creates a profile for a user that includes user preferences, such as the display name and email ad
delete_project	Deletes a project, including project resources
delete_user_profile	Deletes a user profile in AWS CodeStar, including all personal preference data associated with t
describe_project	Describes a project and its resources
describe_user_profile	Describes a user in AWS CodeStar and the user attributes across all projects
disassociate_team_member	Removes a user from a project
list_projects	Lists all projects in AWS CodeStar associated with your AWS account
list_resources	Lists resources associated with a project in AWS CodeStar
list_tags_for_project	Gets the tags for a project
list_team_members	Lists all team members associated with a project
list_user_profiles	Lists all the user profiles configured for your AWS account in AWS CodeStar
tag_project	Adds tags to a project
untag_project	Removes tags from a project
update_project	Updates a project in AWS CodeStar
update_team_member	Updates a team member's attributes in an AWS CodeStar project
update_user_profile	Updates a user's profile in AWS CodeStar

Examples

```
## Not run:
svc <- codestar()
svc$associate_team_member(
  Foo = 123
)

## End(Not run)
```

---

xray *AWS X-Ray*

Description

AWS X-Ray provides APIs for managing debug traces and retrieving service maps and other data created by processing those traces.

Usage

```
xray(config = list())
```

Arguments

`config`	Optional configuration of credentials, endpoint, and/or region.

Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

Service syntax

```
svc <- xray(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

Operations

batch_get_traces	Retrieves a list of traces specified by ID
create_group	Creates a group resource with a name and a filter expression
create_sampling_rule	Creates a rule to control sampling behavior for instrumented applications
delete_group	Deletes a group resource
delete_sampling_rule	Deletes a sampling rule
get_encryption_config	Retrieves the current encryption configuration for X-Ray data
get_group	Retrieves group resource details
get_groups	Retrieves all active group details
get_insight	Retrieves the summary information of an insight
get_insight_events	X-Ray reevaluates insights periodically until they're resolved, and records each intermed
get_insight_impact_graph	Retrieves a service graph structure filtered by the specified insight
get_insight_summaries	Retrieves the summaries of all insights in the specified group matching the provided filte
get_sampling_rules	Retrieves all sampling rules
get_sampling_statistic_summaries	Retrieves information about recent sampling results for all sampling rules
get_sampling_targets	Requests a sampling quota for rules that the service is using to sample requests
get_service_graph	Retrieves a document that describes services that process incoming requests, and downstr
get_time_series_service_statistics	Get an aggregation of service statistics defined by a specific time range
get_trace_graph	Retrieves a service graph for one or more specific trace IDs
get_trace_summaries	Retrieves IDs and annotations for traces available for a specified time frame using an opt
list_tags_for_resource	Returns a list of tags that are applied to the specified AWS X-Ray group or sampling rule

put_encryption_config	Updates the encryption configuration for X-Ray data
put_telemetry_records	Used by the AWS X-Ray daemon to upload telemetry
put_trace_segments	Uploads segment documents to AWS X-Ray
tag_resource	Applies tags to an existing AWS X-Ray group or sampling rule
untag_resource	Removes tags from an AWS X-Ray group or sampling rule
update_group	Updates a group resource
update_sampling_rule	Modifies a sampling rule's configuration

Examples

```
## Not run:
svc <- xray()
svc$batch_get_traces(
  Foo = 123
)

## End(Not run)
```

Index